VLSI DESIGN
2000, Vol. 11, No. 4, pp. 339–351
Reprints available directly from the publisher
Photocopying permitted by license only

A New Bus Assignment Algorithm for a Shared Bus Switch Fabric

D. TORRES*, J. GONZALEZ and M. GUZMAN

Cinvestav del IPN, Research and Advanced Studies Center of National Politechnic Institute, Apartado Postal 31-438, Guadalajara, C.P. 44550, Jalisco, México

(Received 5 June 1999; In final form 10 February 2000)

An architecture for a Cell Switch Fabric (CSF) with a *new bus assignment strategy* is presented. The proposed architecture has a modular structure with a *chip partitioning* oriented to avoid the system from falling down totally, thus achieving expandability and increasing reusability. A discussion about different algorithms for the bus assignment is given. The shared bus is assigned in a *cyclic and rotative way*, switching microcells instead of cells. The CSF is built in four PCBs where every one has a capacity for four ports at 164.323 Mbps, the external port rate is 155.52 Mbps. A microcontroller realizes some tests and communicates with a PC, which runs a test, verification and CSF configuration program. Some parameters about the CSF behavior are measured too. Each Port was implemented on a CPLD FLEX10K100 and the Switch Control Block on a circuit MAX7128, both from Altera Company.

Keywords: Broadband; Switching; Hard-software codesign; Algorithm; Implementation; Testing

1. INTRODUCTION

The Asynchronous Transfer Mode (ATM) has been standardized by the ITU-T, see recommendations I.112-I.751 and Q.2010-Q.2971, as the multiplexing and switching principle for Broadband Integrated Services Digital Network B-ISDN. ATM is a packet and connection-oriented transfer mode based on statistical time division multiplexing techniques, using the philosophy of bandwidth on demand. High speed switches with high performance are very important for broadband networks, which can be based on *routing* or *switching*. A small high speed switch prototype for research and academic purposes – see Figure 1 – is currently being built at our Center. This switch consists of the following modules:

- *a cell switch fabric,*
- *a line circuit for Ethernet and Fast Ethernet Networks* [1, 2] and
- *a line circuit for* 63 *E*1/*T*1 [3].

The kernel every broadband switch is the denominated *Cell Switch Fabric* (CSF), whose

*Corresponding author. Tel.: 52-3-684-1580, Fax: 52-3-684-1708, e-mail: dtorres@gdl.cinvestav.mx

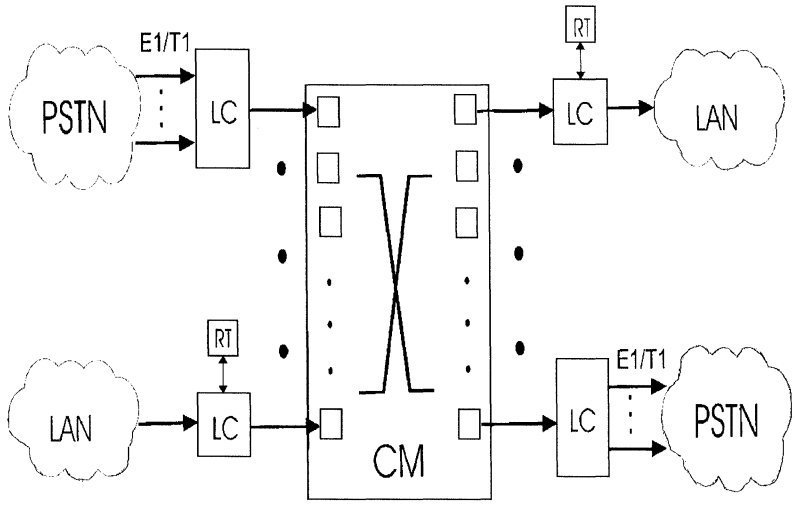

FIGURE 1 A General switch.

design is the major goal of the present work. *Main objectives of this paper are*:

1. to *explain the selected architecture and how it works*,
2. to *design and to implement the CSF on complex CPLD circuits*,
3. to *discuss the switch tests*.

2. CELL SWITCH FABRIC ARCHITECTURE

A switch or the designed cell switch fabric can, in general, be represented as a 7-tuple (N, M, C, P, S, T, f_r), where N is the number of inports, M is the number of outports, C is its capacity, P is the performance – a set of parameters $p_i \in P -$, f_r is the switch *routing function*, S is a set of *cells* and T a set of *tags* related to function.

In a switch the *routing function* f_r can be represented in the following way: given a set of *cells* S and a set of *tags* T, the switch *routes* the cells $s_i \in S$ to the corresponding *output ports* (one or more), *i.e.*, $f_r : S \times T \hookrightarrow \Theta | \Theta \subseteq S$, where every $\theta_i \in \Theta$ is obtained from s_i, also they have the same *information content* $\langle\,\rangle$, or $\forall s_k \in S \Longrightarrow \exists\ \theta_j \in \Theta | \langle s_k \rangle = \langle \theta_j \rangle$ [2].

The design and implementation of a CSF is rather difficult, because it is a *very hard real time system* which must be able to process per port every cell each 2.7 μs. An application of this switch fabric is a Data-Voice Switch – see Figure 1 – or an Ethernet Switch, which can be created if all line card interfaces are Ethernet type [1]. The CSF consists of a maximum of $N = M = 16$, internal transfer rate $V = 164.323$ Mbps – a capacity $C = NV -$.

Architectures [4, 5] based on shared-memory [6, 7] and common bus [8, 9] and their variants [10] have been studied and shown in the literature. To select a specific architecture general design criteria were taken into account including:

- *simplicity*, to achieve a rapid implementation in hardware,
- *expandability*, to achieve modularity and scalability,
- *possibility of implementation on CPLD circuits*, number of gates, number of pins, ...,
- *testability*, every hard- or software component should be testable,
- *costs*, for the economy *and*
- *performance*, because determined *values of* some parameters could be unacceptable in a specific system.

Although these factors are taken into account not all of them have equal weight in the selection of a particular architecture. For example, an architecture can be driven by performance, or by

testability, in other cases by expandability and so on.

The CSF, which is a *new variant* belonging to those based on *common or shared bus*, is composed – see Figure 2 – of the following parts: *Input and Output Elements*, *a Switch Fabric Control Block*, *a Shared Bus and a Test and Supervision Block*. Ports are connected to the Line Circuits using a specific interface (see Section 2.5).

2.1. Input Element (IE)

An IE receives – see Figure 2 – the internal cells of 56 bytes size, stores them in a FIFO *buffer*, divides cells into microcells and delivers the microcells to the shared bus. The buffer management operates according to the classic *producer-consumer* problem. If the buffer is full it sends a wait signal to the associated interface, also, it uses the *backpressure principle*. Although the routing function is carried out into the routing table [2], each IE interprets the information associated with the routing in order to send the cell to the corresponding output port. The IE consists – see Figure 3 – of the following parts:

- *a FIFO buffer*, where cells are stored temporally until its transmission through the shared bus to the corresponding output port,
- *a Write-FSM*, whose function is to accept the input flow and write it in the FIFO *buffer*,
- *a Read-FSM*, which delivers the microcells to the shared bus through the Bus-Buffer,
- *a Bus-Buffer*, this is the interface with the shared Bus, and
- *a Test Interface for the Microcontroller*, which permits the microcontroller to write and read text data in the IE.

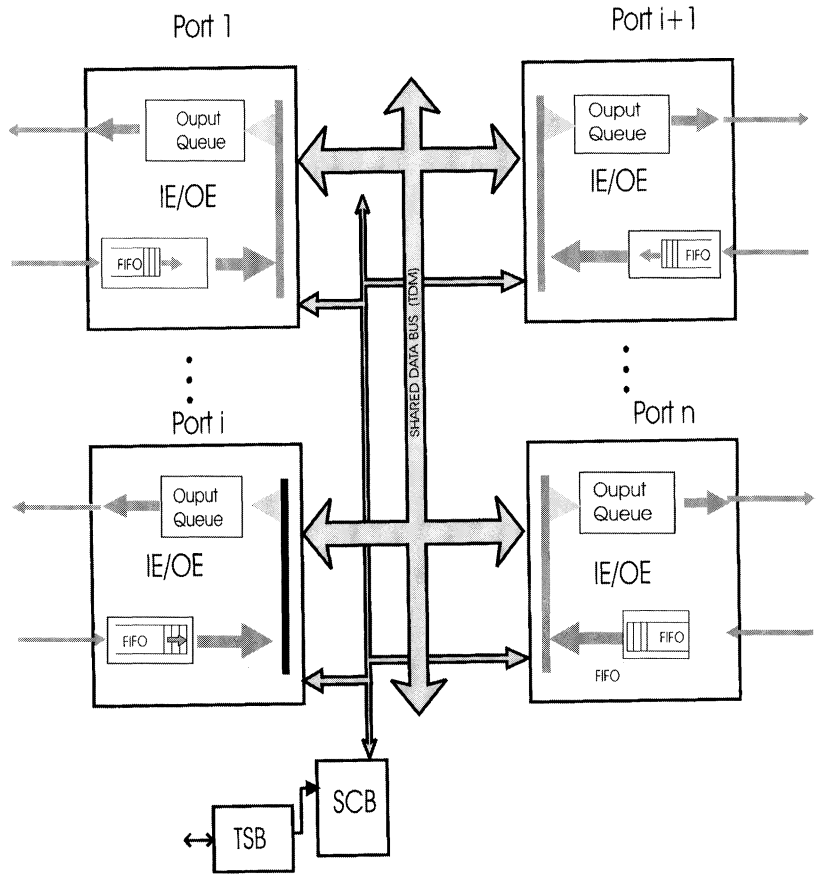

FIGURE 2 Switch architecture fabric.

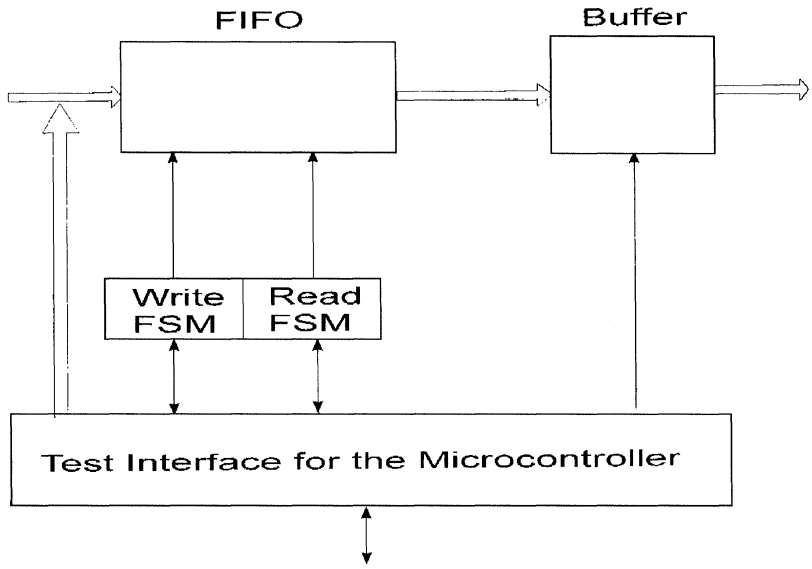

FIGURE 3 Input element.

2.2. Output Element (OE)

An OE receives – see Figure 2 – the microcells from the shared bus, stores them in an output queue, assembles the corresponding cells and sends them to an output interface. To each *active* IE is associated a microcell queue in the OE. For that reason, the complexity of the OE is high. Due to the complexity of the OE special *finite state machines* FSMs [11] were designed to manage them. Additionally OE elements keep information about the *number of lost cells* caused by buffer congestion and the *number of cells* that are waiting in the queue. Therefore these numbers are related with two important *performance parameters*: the *cell loss probability*, P_{loss}, and the *mean waiting time*, τ_W, where P_{loss} and $\tau_W \in P$.

An object oriented conception of this element helped to dominate the design complexity [12]. For example, a buffer is a queue with the basic operations Put(Q_n, x) and Get(Q_n), but for a implementation in hardware these operations are too general. Starting from this point, the functionality of an OE can be seen as a *queue of n cells*, Q_n, where to each cell corresponds a *buffer space*. Any cell can be seen as a *queue of m microcells*, q_m, (in our case $m = 7$). Then the total number of queues is in our case, $m \times n$. Consequently, the number of read/write pointers to manage these queues is excessively large. The mentioned two basic operations take the form:

Put(Q_n, i, j, x), stores a microcell x, into the queue Q_n at the position j of the cell buffer space i,

Get(Q_n, i, j), returns a microcell x from the queue Q_n at the position j of the cell buffer space i and sends it through the interface to the associated line circuit,

These functions are more specific to implement than their ancestors. In order to avoid a laborious implementation with $2 \times m \times n$ pointers, we used only two *reloadable pointers* (one for read and one for write operation), which were implemented by FSMs. This type of pointer should be initialized with (i, j) before each operation, but simultaneously it must keep its old content, because this value will be used later to calculate a new pair (i, j). In this way, we can control the successive write or read operations to specific memory locations.

DEFINITION 1 A **reloadable pointer** is that which can be initialized before or after each operation, but simultaneously it must keep its old content (an address), for later use.

An OE includes – see Figure 4 – the following parts:

- a *Bus-Buffer*, from where microcells are taken to be stored in the Output-Queue,
- an *Output-Queue*, it is a queue of cells in which each cell is associated with a queue of microcells,
- a *Write-FSM*, which takes microcells from the Bus-Buffer and stores them in the corresponding position of the associated queue,
- a *Read-FSM*, whose main function is to send a cell, stored in the Output-Queue, to the Interface Line Circuit,
- an *Interface to Line Circuit*, to match the output port with the SCF and
- a *Test Interface for the Microcontroller*, which permits to the microcontroller write and read data in the OE.

2.3. Switch Fabric Control Block (SCB)

It generates the system clocks and it synchronizes the bus rotative cyclic assignment for the microcell transfer between the input and output elements, also *this bus assignment strategy is implemented in this block* and it will be explained in details below. The main disadvantage of this method is the assignment of the bus to input elements that perhaps do not have, at this moment, a microcell. For this reason this strategy does not provide the most efficient use of the shared bus.

2.4. Test and Supervision Block (TSB)

The core of this block is a microcontroller interfaced with the other different blocks and which can communicate with a PC, *via* an interface RS232C, by means of a developed protocol (see Section 4.1.5). The microcontroller program has a volume of 6Kbyte, which is stored in its EEPROM.

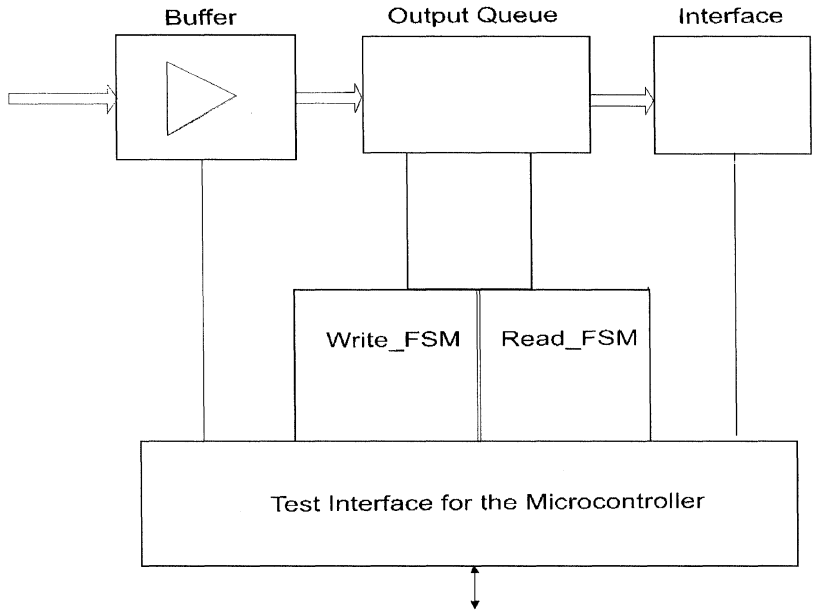

FIGURE 4 Output element.

2.5. Line Interface Circuits (LIC)

Ports of the SCF can be reached through an interface developed in our center for this purpose, although with a few changes the interface UTOPIA Level 1 or 2 can be implemented, that consists, in the *reception direction*, of the following signals:

- eight data bits for reception,
- buffer full signal,
- reception clock

and in the *transmission direction* of:

- eight data bits for transmission,
- active cell signal,
- transmission clock.

3. SCHEDULER ALGORITHMS

The cell switch fabric has N inports and N outports, through the inports, implemented by IEs, N concurrent and unsynchronized cell data streams come into itself, through the N outports, implemented by OEs, N concurrent cell data streams come out. We denote the input streams by ID_k, and the output streams by OD_k, $k \in \{1, 2, \ldots, N\}$ see Figure 5. All the input streams flow into a *common bus* which is *a shared resource* by all of them. In order to share the bus, a scheduler to assign it to the input streams is required [13, 14]. Two requirements for the bus scheduling are: *low cell latency* and *fair assignment* so that a high traffic data stream does not hog the bus or buffers to the detriment of the other data streams.

Shared Bus (SB) is composed of a 64 bits width data path, and a control path. Through the control path the bus scheduling algorithm assigns the data path. The control path is used to select input and output ports, which are connected during a time slot. On each time slot, data from the input stream flows to the output stream. There is a number of

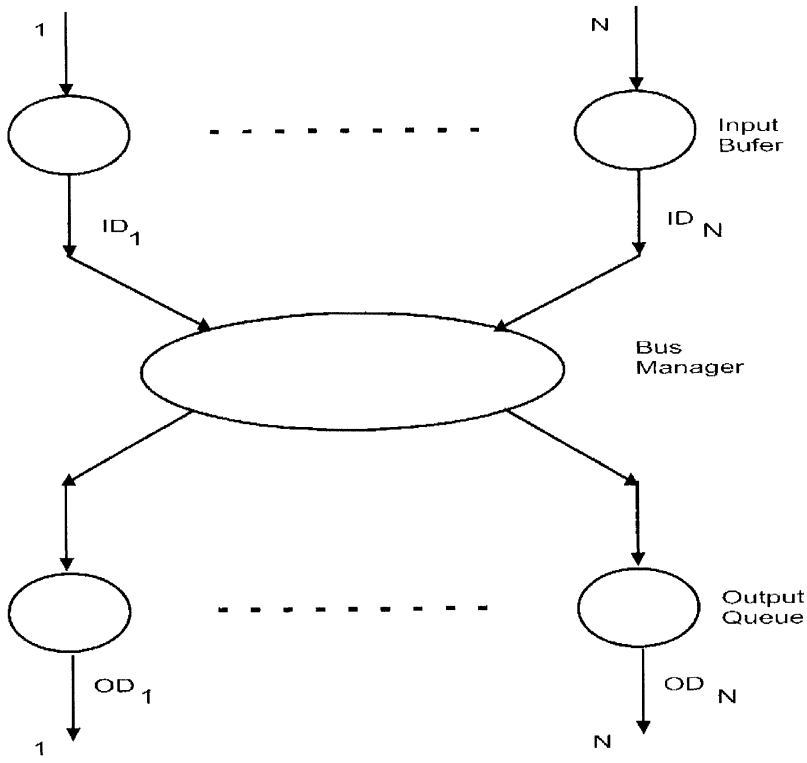

FIGURE 5 Concurrent data streams ID/OD.

fashions to assign the bus to transfer data between input and output ports. In general, the bus can be assigned in a static or a dynamic way. The static assignment algorithms waste time slots when the bus is assigned to a port, which has no data to transmit. On the other hand, this does not happen in dynamic assignment algorithms, which only assign the bus to input ports, which have data to send. From the *circuiting complexity point of view static, in general, are simpler than dynamic algorithms*. In what follows, we consider two static and one dynamic bus assignment algorithms namely:

- cyclic to all IEs (static),
- rotative cyclic to all IEs (static),
- dynamic and sequential to all *active* IEs (dynamic).

Before explaining in more detail the bus assignment policies *some concepts*, which are helpful to understand the description are addressed. In what follows, it says that an input or output element is *active*, if it has a cell to be switched. The shared bus works in a *cyclic way* distributing time quantum q_i or time slots to each port. The SCF switches *microcells* instead of *cells*. Therefore *cells* are divided into *microcells*, a cell has k $(=7)$ microcells, and each microcell has 64 bits. *Microcell time slot* τ has a duration equal to a microcell duration, *i.e.*, $\tau = 22.91$ ns. A *frame* has a duration $\gamma = (N+1) * \tau$, where N is the number of ports, for $N = 16$, $\gamma = 389.47$ ns. Several frames, j $(=7)$, build a *multiframe*. During the time interval denominated *bus cycle* ϕ is carried out the transmission of a multiframe, a bus cycle ϕ has a duration of 2.7263 μs. All these concepts are interleaved according to the Figure 6.

3.1. Cyclic Assignment

The cyclic assignment is the simplest form, and it can be described in the following way.

PROPOSITION 1 *The output function m_i to select one of the N IEs can be, in general, given by*:

$$m_i = i \bmod N$$

where $i \in \{0, 1, \ldots\}$ is the number of the time slot; $m_i \in \{0, 1, \ldots, N-1\}$ and N is the number of ports.

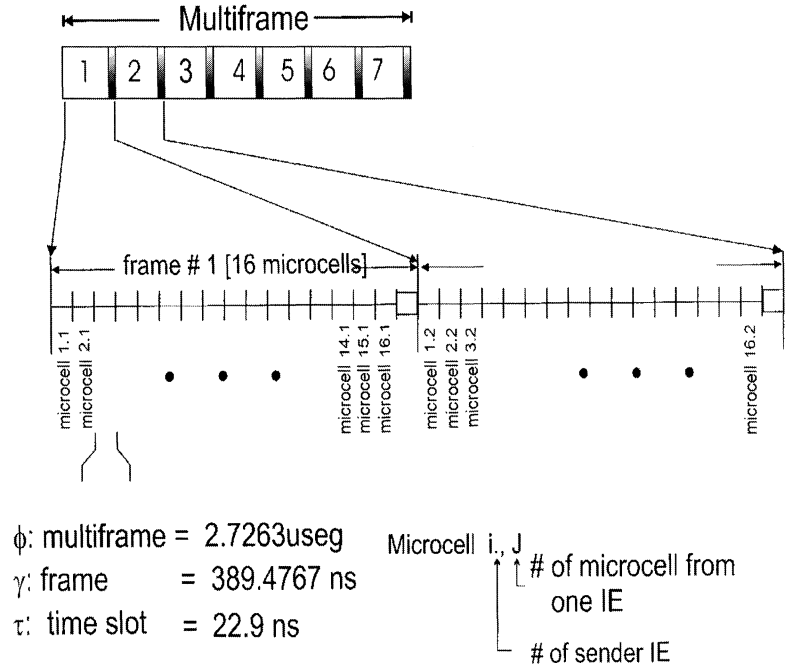

FIGURE 6 Multiframe format.

Proof During each time slot i a number between 0 and $N-1$ is generated and it is given by i mod N. Then the inports are selected sequentially and always with the same sequence. ■

This form constitutes the easiest implementation, but it does not take into account that not all I D_k are active simultaneously and therefore waste time slots. The *main disadvantages* of this assignment policy are: the bus can be assigned to an IE, which is not active and, if there is an OE space for one cell, the probability of success is not equal for all IEs. From the traffic point of view the cyclic assignment leads to the following: as each port has an *index*, given by a number from $0, \ldots, N-1$, those ports with a low index have a probability to find an empty cell buffer space in the destined OE slightly greater than the others. Strictly speaking, the cell loss probability depends on the port index.

3.2. Rotative Cyclic Assignment

For this algorithm, the control block introduces specific sequences. In the proposed SCF, the bus is assigned not only in *cyclic* form but also in a *rotative* form. In this way, if many cells arrive simultaneously to the switch fabric, the *probability of the selection of an IE does not depend from its position in the switch*. Therefore, the *probability to find an empty cell buffer space in the destined OE is equal for all input ports*. Also, this algorithm avoids the unfairness present in the cyclic assignment and its circuiting complexity is slightly higher than the one in the cyclic assignment. Time slots are assigned through a *round-robin algorithm*, where the *time-quantum* $q = \tau$.

PROPOSITION 2 *The output function* m_{ik} *to select one of the N IEs can, in general, be given by*:

$$m_{ik} = 1 + (i + k) \ mod \ N(2)$$

where i is the time slot number, k is the frame number, and N is the number of ports.

Proof *If* the number of time slots -time interval of a microcell – $i \in \{0, 1, \ldots, \}$, the frame number $k \in \{0, 1, \ldots, \}$ *then* $(i + k)$ mod N generate sequences of the form $< 0,1,2, \ldots, (N-1) >$; $< 1, 2, \ldots, (N-1), 0 >$; $< 2, \ldots, (N-1), 1, 2 >, \ldots$ ■

3.3. Dynamic and Sequential Assignment

The principle of dynamic assignment is to distribute time only between the *active inports*. This is the most complex form, because it is necessary:

1. to generate the time quantum q_i,
2. to know which IEs are active, and
3. to assign dynamically the quantum q only to the active IEs in sequential order and to start with the one that has smallest index.

PROPOSITION 3 *The output function* m_j *to select one of the I active IEs and to assign it a time quantum can, in general, be given by*:

$$m_j = f(\langle b_{j=i \ mod \ l} \rangle)$$

where $m_j \in \{0, 1, \ldots, l\}$, *l is the number of active ports*, $l \leq N$, *N is the number of physical ports and* $\langle b_j \rangle$ *is the content of* b_j.

Proof Each time quantum q_i is created according to the equation $q_i = i$ mod l, where $i \in \{0, 1, \ldots\}$; and l is a *variable*, which represents the number of *active* IEs. To every IE_k or ID_k is associated a pair (c_k, a_k), where c_k is an identifier, which take the value $c_k = 1$ if the *inport is active*, and where a_k is the inport address. A static or dynamic *array* b_j *containing the addresses of the active inports* can be used to keep the port addresses. Then b_j is a pointer array to the inport numbers, $\langle b_j \rangle = a_k$, where the subindexes j and k are independent each other. For the cell arrival sequence given by $j1, j2, \ldots, jl$ the content $\langle b_1 b_2 \ldots b_l \rangle = a_{j1} a_{j2} \ldots a_{jl}$, generating the expected sequence. Starting of a_k is selected the inport m_j, therefore $m_j = f(\langle b_{j=i \bmod l} \rangle)$. Consequently the implementation of this algorithm is laborious. ■

Taking into consideration all the facts described the *rotative cyclic algorithm* was chosen to assign the bus of the implemented switch fabric. The CSF capacity is about 2.5 Gbps but the *throughput* is a linear function of *the number of active ports* and the *time efficiency on the bus* is a function of the bus assignment algorithm. Simulations about this question are under study. The behavior of these algorithms can be described using a programming language for simulation studies.

4. SCF IMPLEMENTATION AND TESTS

Telecommunication digital systems are, today, generally programmable and consequently they are built of hardware and software components. It is very convenient and important the reusability of these components, because the complexity of circuits and systems increases continuously. To master this problem the introduction of CAD tools for hard-/software co-design is decisive. Although there are important and useful results as well as developed tools to help designers in the *modeling*, *validation* and *synthesis*; they are not enough. Due to all these factors, many researchers are motivated to study new scientific problems and to develop new methods to solve classic problems [13–15]. Attempts to characterize digital systems are given in the literature [13] chapter 1, using notions such as: *application domain*, *degree of programmability*, *hardware implementation technology and level of integration*.

This work deals with a SCF prototype to validate a design, using CPLDs, a microprocessor, internal and external memories, and software components. *System-level partitioning* [16] into hard- and software components is a very important task in order to achieve failure-tolerance, testability, scalability and reliability among other factors. *Identification of critical segments of code* to assure performance on the programmable hardware was and is a very important task. Functions were implemented using FSMs and interconnection of some of them. For the SCF implementation the *rotative cyclic bus assignment* strategy was selected. In addition to the above mentioned factors the following requirements were decisive for the selection of the architecture: *the bandwidth*, the *internal parallelization level* on the shared bus $L = 64$ bits, *memory needs* for the temporal storing of the cells. *Major characteristics of* our design are – see Figure 7:

- Architecture with four PCBs and each PCB card has four CPLDs.
- Each port was implemented on a CPLD FLEX10K100 [17, 18], occupying 211 pins, 50% of logic resources and 67% memory capacity.
- The Switch Fabric Control Block was implemented on a circuit MAX7128.
- The shared bus implementation is a hard problem due to the high operation frequency and the appearance of its harmonics.
- Two or more PCB cards can be connected together to enhance the CSF bandwidth.
- Partition of the hardware and software and their concurrent design.
- Use of a microcontroller on board for some tests and for continuous monitoring.

4.1. Switch Tests

The importance of test methodologies grows continuously due to the increasing circuit complexity. An example of them is the Built-In Self Test (BIST), in which test functions are embedded into the circuit itself [19]. For the SCF was implemented a Test and Supervision Block TSB, which realizes the following functions:

- *Autotest*,
- *Input element test*,
- *Output element test*,
- *Bus test*,
- *Switch configuration*,
- *Statistic control of certain parameters*,
- *Communication with a PC*.

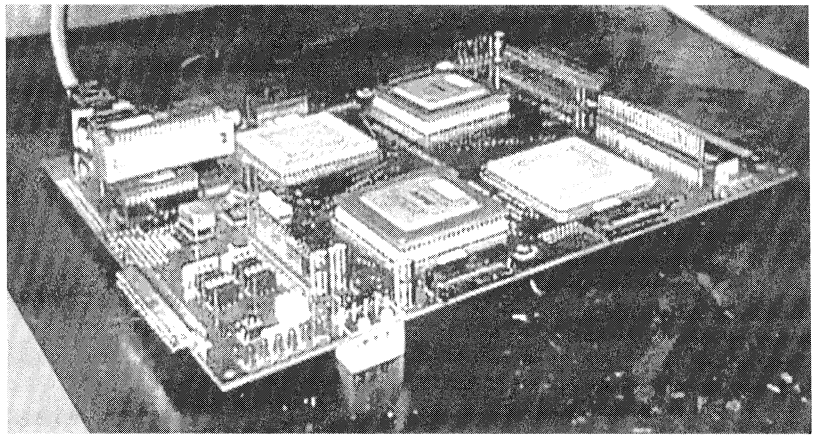

FIGURE 7 PCB card for a SCF with four ports.

Reference [20] provides details about testing. Some basic rules for telecommunication systems design used, in this work, are the following:

- isolation at different levels,
- normal access isolation as required,
- recovery strategy,
- observability during system operation,
- on-board controller.

4.1.1. Autotest

By power on, an exhaustive switch test is performed automatically. As a result of this test, each port is ready or out of order. In the last case, the switch works only partially, also it suffers a degradation but the switch does not fall down totally.

4.1.2. Input Element and Output Element Test

As these blocks contain FIFO memories and many finite state machines of different complexity, data are written in and read from the memories. In addition the FSMs are tested with the help of prefixed test sequences. By means of write and read operations into the memories the shared bus is tested partially.

4.1.3. Switch Configuration

The switch can be configurated to operate with a clock at:

- 43.8 MHz/ 16 ports/ 2.5 Gbps,
- 21.9 MHz/ 8 ports/ 1.25 Gbps,
- 10.95 MHz/ 4 ports/ 622 Mbps.

4.1.4. Statistics Control

A statistics control about the switch behavior is carried out to keep the corresponding information updated. Two performance parameters of the set P are measured into the SCF:

- the *number of lost cells in the output element*, related with P_{loss}, and
- the *number of cells waiting in the output element*, related with τ_W.

4.1.5. Communication Protocol

The communication mechanism between the microcontroller and the PC must be reliable and robust among other characteristics. For this purpose, a communication protocol was developed, and implemented [21]. In a PC, the program runs under Windows and was written using the object-oriented programming and $C++$ language, while

the corresponding program for the microcontroller was written in *C*. The protocol has the following characteristics:

- *asynchronous*,
- *serial at different transmission rates*: from 1.2 Kbps to 38.4 Kbps, higher transmission rates are not necessary,
- *frames variable length* from 5 to 255 bytes, to avoid packets too large and to check them rapidly,
- *control fields of these frames with an orthogonal format*, to extract and to process the fields efficiently,
- *check bits* using the polynomial $x^{16}+x^{15}+x^2+1$, to detect and to correct transmission errors,
- *sequence control per frame*, to detect if a packet was lost, in this case part "*A*" requests part "*B*" a retransmission,
- *frame acknowledgment*, to inform the other part about the correctness of the received frame,
- data is never lost.

Main objective the PC-Microcontroller communication is to establish a dialog between them:

- to request test on each input and output elements, testing external and internal memories and FSMs, among others elements,
- to get information about the mentioned performance parameters,
- to change the switch configuration.

5. CONCLUSIONS AND REMARKS

An efficient broadband switching system design is, today, an important challenge in telecommunication area. In this paper, we conclude:

- The architecture of the CSF, which is a *new variant* belonging to those based on *common or shared bus*, is composed of the following parts: Input and Output Elements, Switch Fabric Control Block, Shared Bus and Test and Supervision Block.
- The architecture has a modular structure with a *chip partitioning* oriented to avoid that the system fall down totally, to achieve expandability and to increase reusability.
- Different bus assignment algorithms were discussed in this paper.
- A rotative cyclic bus assignment was implemented, where cells are divided into microcells. Microcells are switched through the shared bus.
- The SCF is built in four PCBs where every one has a capacity for four ports at 164.323 Mbps, this corresponds to an external port rate of 155.52 Mbps. The rates difference is due to the size difference between ATM cells (53 Bytes) and internal cells (56 Bytes).
- Tests are performed by the corresponding circuit, a microcontroller interfaced with the elements of the switch fabric.
- By power on, the switch fabric is fully tested and in the case of an error other routines are activated in order to put out of service the associated circuit. In this way, the switch fabric suffers only a degradation but not a total failure.
- The microcontroller can communicate with a PC, in which runs a test, verification and SCF configuration program. It can switch the bus operation frequency and measures some parameters about the CSF behavior.
- Each port was implemented on a CPLD FLEX10K100 and the control block SCB on a circuit MAX7128, both from Altera Company.
- Due to the high operation frequency and the appearance of its harmonics, the common bus design on the PCB is problematic. Therefore, it is necessary to take special hand craft.

Remarks In what follows, some remarks will be given based on our experience:

- It is useful the decomposition of the systems in modules, which contain simple and compound components.

- Not only for the software but also for the hardware it must to work for the reuse, trying to find components, which can be used later in other designs.
- Hard- or software components must be testable.
- Some tests must be embedded in the circuit, built-in self-test methodology.
- Components should be continuously improved to be reused in other designs.
- Complex circuits must have interfaces to be tested, standards interfaces has been developed for this purpose.

Acknowledgments

The authors wish to express their gratitude to the Semiconductor Technology Center for the technical support given to this work.

References

[1] Medina, R., Torres, D. and Guzman, M., An Interface for a Fast Ethernet LAN Network, *Proc. of the International Symposium on Information Theory and Its Applications*, ISITA 98, pp. 564–567.
[2] Torres, D., Larios, A. and Guzman, M., Routing Chip based on a modified trie for ATM, IP and Ethernet: Hard/Software Co-Design, *Proc. of IEEE ISCASS'99*, **I**, 427–430, Orlando, USA.
[3] Lopez, G., Torres, D. and Silva, J., A Line Interface Circuit for 63 E1/T1, In: *Development*.
[4] Tobagi, F. A., Fast Packet Switch Architectures For Broadband Integrated Services Digital Networks, In: *Proc. IEEE*, **78**(1), 133–167, Jan., 1990.
[5] Acampora, A., An Introduction to Broadband Networks: LANs, MANs, ATM, B-ISDN, and optical networks for integrated multimedia telecommunications: Plenum Press, New York, 1994, pp. 146–148.
[6] Yasuro, S. *et al.*, "A One-Chip Scalable 8×8 ATM Switch LSI Employing Shared Buffer Architecture", *IEEE J.S.A.C.*, **9**(8), Oct., 1991, 1248–1254.
[7] Barri, P. and Goubert, J. A., "Implementation of a 16 to 16 Switch Element for ATM Exchanges", *IEEE J.S.A.C.*, **9**(5), June, 1991, pp. 751–757.
[8] Shiro, K. and Naoaki, Y., "An Expandable Time Division Circuit Switching LSI and Network Architecture for Broadband ISDN", *IEEE J.S.A.C.*, **14**(2), February, 1996.
[9] Jajszcyk, A. and Tyszer, J., "Broadband Time-Divison Circuit Switching", *IEEE J.S.A.C.*, **14**(2), Feb., 1996.
[10] Plaza, P., Merayo, L. A. *et al.*, A 2.5 Gb/s ATM Switch Chip Set, *IEEE Trans. on VLSI Systems*, **4**(3), Sep., 1996, 405–416.
[11] Kam, T., Villa, T., Brayton, R. and Sangiovanni, A., *Synthesis od Finite State Machines: Functional Optimization*, Kluwe Academic Publishers, First edn. 1997.
[12] Heilemann, G. L. (1997). *Data Structures, Algorithms, and Object-Oriented Programming*, McGraw Hill, Inc.
[13] De Micheli, G. and Sami, M., *Hardware/Software Co-Design*, Kluwer Academic Publishers, First edn. 1996.
[14] Bolsens, I. *et al.*, Hardware/Software Codesign of Digital Telecommunication Systems, *Proc. of the IEEE*, March, 1997, pp. 391–418.
[15] De Micheli, G. and Gupta, R. K., Hardware/Software Co-Design, *Proc. of the IEEE*, March, 1997, pp. 349–365.
[16] Staunstrup, J. and Wolf, W., *Hardware/Software Co-Design*, Principles and Practice, Kluwer Academic Press, First edn. 1997.
[17] Leach, T. and Hackett, B., "*Modern Synchronous Design*", ASIC and EDA, January, 1993, pp. 44–52.
[18] Stephen, W., "*Design Tips for High-Performance FPGA Design*", ASIC and EDA, October, 1994, pp. 41–52.
[19] Mukherjee, N. and Chakraborty, J., Built-In Self-Test: A Complete Test Solution for Telecommunication Systems, *IEEE Communications Magazine*, **37**(6), June, 99, 72–78.
[20] Adham, S. M. I. *et al.*, Linking Diagnostic Software to Hardware Self-Test in Telecom Systems, *IEEE Communications Magazine*, **37**(6), June, 99, 79–83.
[21] Torres, D., Hermosillo, J., Figueroa, M., Silva, J. and García, A., Development and Implementation of a Communication Protocol for Voice and Data Applications, in Spanish, *Proc. of ELECTRO 98*, Chihuahua, 1998, pp. 93–98.

Authors' Biographies

Deni Torres-Roman (dtorres@gdl.cinvestav.mx) received a Ph.D. degree in telecommunication from Technical University Dresden, Germany in 1986. He was professor in University of Oriente, Cuba. He received the Telecommunication Research Prize in 1993 from AHCIET Association and was recipient of the 1995 Best PaperAward from AHCIET Review, Spain. Coauthor of a book about Data Transmission. Since 1996 he is associate professor at Center for Research and Advanced Studies of IPN (CINVESTAV-GDL) Mexico. His research interests include VLSI, hard- and software designs for telecommunication area. He is a member of the IEEE.

Jesus Gonzalez-Pintor (jesusg@level1.com) received a M.Sc. degree in telecommunication from CINVESTAV-GDL Mexico in 1998. Currently he is an engineer in TDCOM, Mexico. His research interests include VLSI designs. He is a member of the IEEE.

Manuel Guzman (manuelg@gdl.cinvestav.mx) received the Ph.D. degree in Electrical Engineering from the Royal Institute of Technology, Stockolm,

Sweden. His main research interests are in the hard/software codesign for digital circuits and algorithm implementation on VLSI. He is currently an Associate Professor at the Centre for Research and Advanced Studies (Cinvestav) of IPN, Guadalajara, Mexico and Head of the Semiconductor Technology Center. He is a member of the IEEE.